A CNN based Approach for Handwritten Character Identification of Telugu Guninthalu using Various Optimizers

B. Soujanya[1]
Assistant Professor, Dept of Computer Science and Engineering
Institute of Technology, GITAM (Deemed to be University, Visakhapatnam, India

Suresh Chittineni[2], T. Sitamahalakshmi[3]
Professor, Dept of Computer Science and Engineering
Institute of Technology, GITAM (Deemed to be University
Visakhapatnam, India

G. Srinivas[4]
Associate Professor, Dept of Computer Science and Engineering, Institute of Technology, GITAM (Deemed to be University, Visakhapatnam, India

***Abstract*—Handwritten character recognition is the most critical and challenging area of research in image processing. A computer's ability to detect handwriting input from various original sources, such as paper documents, images, touch screens, and other online and offline devices, may be classified as this recognition. Identifying handwriting in Indian languages like Hindi, Tamil, Telugu, and Kannada has gotten less attention than in other languages like English and Asian dialects like Japanese and Chinese. Adaptive Moment Estimation (ADAM), Root Mean Square Propagation (RMSProp) and Stochastic Gradient Descent (SGD) optimization methods employed in a Convolution Neural Network (CNN) have produced good recognition, accuracy, and training and classification times for Telugu handwritten character recognition. It's possible to overcome the limitations of classic machine learning methods using CNN. We used numerous handwritten Telugu guninthalu as input to construct our own data set used in our proposed model. Comparatively, the RMSprop optimizer outperforms ADAM and SGD optimizer by 94.26%.**

Keywords—Character recognition; Adam; RMSProp; SGD; CNN

I. Introduction

In today's world, the internet is brimming with images and video representations, providing sufficient opportunity for building numerous research applications for image and video analysis [1] to educate people about more complex material and techniques. With the rise of Artificial Neural Networks, machine learning has advanced significantly in recent years (ANN). These ideas enhance the model's capabilities beyond machine-learning tasks and other domains. Convolutional neural network (CNN) architecture has been considered as one of the most inventive. Using CNN in image processing became clearer and more beneficial as ANN performance deteriorated in object recognition and image classification. As better CNN became accessible, research using CNN in image processing domains grew dramatically [2-4]. CNNs have had a lot of success in various domains, including computer vision, natural language processing, and speech recognition.

One of the most widely used machine learning models is CNN which has been expanded to handle a wide range of visual image applications, item classification, and audio identification challenges by applying mathematical representations. Multi-layer network structure that may be learned and consists of several layers [5]. Raw pixel values may be utilized as input to the network instead of feature vectors, which are often employed in machine learning. Even though there are many different kinds of CNNs (fig.1), they always have the same basic structure: a convolutional layer, a pooling layer, and an entirely connected layer.

1) Convolution layer: Images are filtered using this tool, which identifies characteristics that are used to identify matching spots during testing. Enlarged images need a convolution procedure with minimum parameters. With a filter or kernel, the input data is transformed into a feature map for use by CNN.

2) Pooling layer: This layer receives the extracted characteristics. It reduces bigger images while keeping the most critical data. It keeps the maximum value from each window by preserving the best fit value. This function shrinks the picture spatially to minimize the number of parameters and computations in the model. Max Pooling is a typical strategy in pooling. It selects the greatest element from the feature map covered by the filter.

3) Fully connected layer: High-level filtered images are fed in and categorized using labels in the final layer. Every neuron in this layer is related to the one below it. Layers of convolution and pooling are common in most designs.

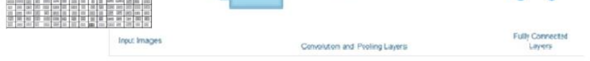


Fig. 1. Layers CNN.

II. Literature Review

CNN performs well in various applications, inspiring academics to work on it in key fields such as natural language processing (NLP), image classification and face recognition, predictive analytics, and so on. P.V. Ramana Murthy et al. [6] built a model that recognizes online handwritten Telugu letters for various domains and companies, yielding a model with 98.3 per cent accuracy, exceeding their expectations. P. Sujatha et al. [7] used CNN architecture to identify a few deep learning strategies for detecting Telugu and Hindi scripts. They also developed a new architecture for identifying low-level textual properties of handwritten characters. Buddharaju Revathi et al. [8] also conducted a survey on Optical Character Recognition (OCR) for the Telugu language, demonstrating the progress of processing the characters stage by stage and performing operations such as segmentation and processing, which resulted in higher accuracy.

B. Hari Kumar et al. [9] used data from several sources to conduct script identification in Telugu. Konkimalla Chandra Prakash et al. [10] used CNN for Telugu script recognition. They supplied a list of Telugu typefaces, a client-server solution for the algorithm's online deployment, and deep learning-based OCR techniques in their study. The segmentation method can be improved so that each character, along with its gunintham and vattu, is segmented. Chirag I Patel et al. [11] developed a technique for identifying characters in a given digitalized text and reading the changing effects of the Models utilizing ANN by employing a back growing neural network to improve script identification accuracy. A.Ram Bharadwaj et al. [12] built a model for Telugu text extraction and recognition using CNN and a recurrent neural network (RNN) with their own data set, achieving an accuracy of 81% by selecting 100 random words from the validation set.

III. Proposed Method

In image-based classification, the CNN architecture is a popular choice. To distinguish features like edges and forms, CNNs apply a range of filters composed of trainable parameters to an image. These high-level filters often utilize weights learned from the spatial attributes of each subsequent level to capture an image's spatial properties. For this design, parameters named as Hyper Parameters are employed. These parameters may influence both the network architecture and training before the training begins.

Hyper Parameters used for building the network are:

- Count the number of hidden layers: the layers between input and output.
- To prevent overfitting, dropouts are employed.
- The activation function is used to add some nonlinearity.
- Learning Rate: It specifies the rate at which the network's parameters are updated.
- Momentum: It keeps things from oscillations.
- One complete prediction cycle of a neural network is one epoch.

Batch Size: The number of subsamples sent to the network.

Fig. 2 [13] depicts the suggested model's implementation stages.

To speed up CNN model training, all of the images in the build data set were categorized, reduced to 70 by 70 pixels, transformed to greyscale, and saved in .npz format. Telugu Guninthalu dataset was trained using three hidden layers of a CNN and tested on it. The network consists of three convolutional layers, two Maxpooling layers, and a fully connected layer. The first hidden layer is a Convolution2D layer. A Maxpooling layer with a pool size of 22 was then employed, along with 256 filters, each with a kernel size of 33 and a stride of 1. A layer of padding has been added to the design to keep the original input size.

Each of the 256 convolution filters in the second convolution layer has a kernel size of 33, followed by a Maxpooling layer of 22. New features have been added to the algorithm, including an additional 256-filter convolution layer and a Maxpooling layer of pool size 22. The two-dimensional matrix data is first transformed into a one-dimensional vector using a Flatten layer before the fully connected layers are generated. Next, we used a totally corresponding layer with the activation of ReLu in it. Dropout, a regularization layer, is set to randomly eliminate 20% of the layer neurons to prevent overfitting.

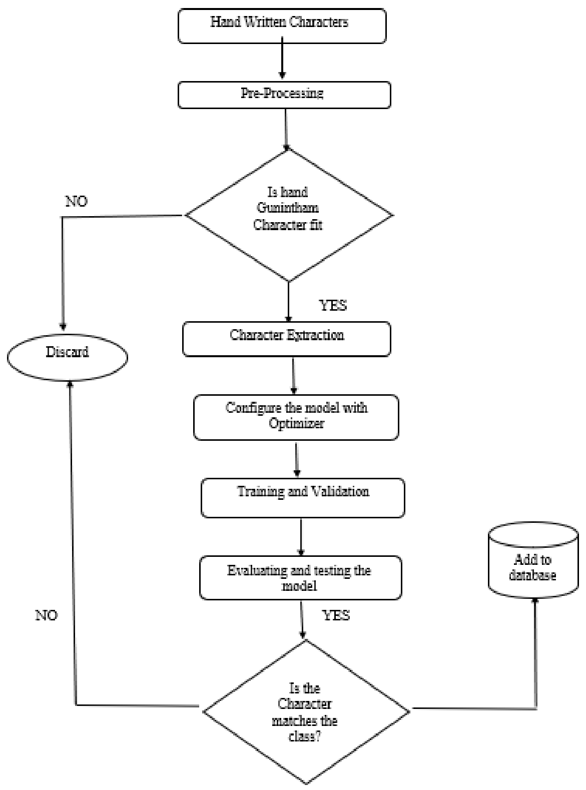

Hand Written Characters
Pre-Processing
Is hand Gunintham Character fit
NO
YES
Discard
Character Extraction
Configure the model with Optimizer
Training and Validation
Evaluating and testing the model
Add to database
NO
YES
Is the Character matches the class?

Fig. 2. Process Flow.

At long last, a sigmoid activation function is added to the 16-neuron output layer. An essential part of CNN is the activation function; it determines the output of a neuron concerning a set of inputs. The activation function is used to introduce nonlinearity into the model. A CNN model's performance is improved by selecting the appropriate activation function. The ReLu and Sigmoid activation functions are used in the suggested model.

1) Rectified Linear Unit (ReLu): It's a biologically and mathematically sound activation function [14]. If the input is negative, ReLu returns 0, if the input is positive; it returns the value itself (Eq 1). ReLu's max operation calculates more quickly than other activation functions. For many kinds of neural networks, it is the default activation function.

$$f(x) = max(0, x) \tag{1}$$

If we are doing a matrix-vector product, this is often done element by element to get the desired outcome. Nonlinearity is vital for CNN since every layer in the network contributes to it.

2) Sigmoid: Eq 2 indicates that it is a probabilistic strategy for decision-making with a range of 0 to 1. We used this activation function to forecast an output since it would be more accurate.

$$f(x) \;=\; 1/\left(1+e^{-x}\right) \tag{2}$$

It's beneficial for models that forecast probability as a result. Because the probability of anything spans between 0 and 1, a differentiable sigmoid function is the best option. As a result, this function yields the curve's slope at any two locations.

Training and validation errors were calculated after each cycle. The training is completed when the number of epochs in training and validation hasn't changed considerably. On the test set, faults in training and validation were discovered, and the network was approximated. It is vital to utilize optimizers in order to improve accuracy and decrease mistakes. Optimizers alter the weight settings in order to minimize the loss function. To get the greatest possible design, we considered a set of priorities or limitations. Adaptive Moment Estimation (Adam), Root Mean Square Propagation (RMSprop) and Stochastic Gradient Descent (SGD) were used in the proposed model.

3) Adaptive Moment Estimation (Adam): Using Adam, an adaptive learning rate technique, individual learning rates are calculated. The exponentially declining average of the preceding squared gradient is stored, and the value of previous historical gradients is preserved, much as momentum. It may also be used to replace stochastic gradient descent systems when updating the network weights in training data. Equations 3 and 4 are used to derive the gradients, which are then used to estimate the moments using exponentially moving averages. Where m_t and v_t are moving averages, g is gradient, β1, β2, and second moment of gradients are gradient forgetting features, and index t is the current training iteration.

$$m_t = \beta_1 m_{t-1} + (1-\beta_1)\, g_t \tag{3}$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2)\, g_t^2 \tag{4}$$

The recommended Adam configuration parameters [15] of β_1=0.9 and β_2=0.999.

4) Root Mean Square Propagation (RMSprop): This approach is similar to a gradient descent with the momentum that confines oscillations to the vertical plane. Thus, increasing the learning rate enables the algorithm to make considerable horizontal jumps toward rapid convergence. The magnitude of recent gradient descents is used to normalize the gradient. Selecting alternative learning rates for each parameter causes the pace of learning to be automatically changed. The parameters are updated using Eq. 5, 6, where gt is the gradient at time t, vt is the exponential average of squares of gradients, and η is the learning rate, which is set at 0.001.

$$v_t = \rho v_{t-1} + (1-\upsilon)\, g_t^2 \tag{5}$$

$$\Delta\omega_t = -\frac{\eta}{\sqrt{v_t + \varepsilon}}\, g_t \tag{6}$$

5) Stochastic gradient descent: Stochastic Gradient Descent (SGD) is a simple yet efficient optimization algorithm used to find the parameters/coefficients of functions that minimize a cost function. In other words, it is used for discriminative learning of linear classifiers under convex loss functions such as SVM and Logistic regression.

$$E(w,b) = \frac{1}{n}\sum_{i=1}^{n} L\left(yi, f(xi)\right) + \alpha R(w) \tag{7}$$

IV. Experimental Results

A. Data Set

Due to the lack of publicly available training data for Telugu characters, we had to construct our own dataset. There is an initial gathering of people's handwritten Telugu guninthalu in various formats and scanning them into precise symbols. Fig. 3 shows a scanned copy of a handwritten guninthalu both online and offline. There are 16 characters in each of the 21 guninthalu, for a total of 275520 handwritten characters.

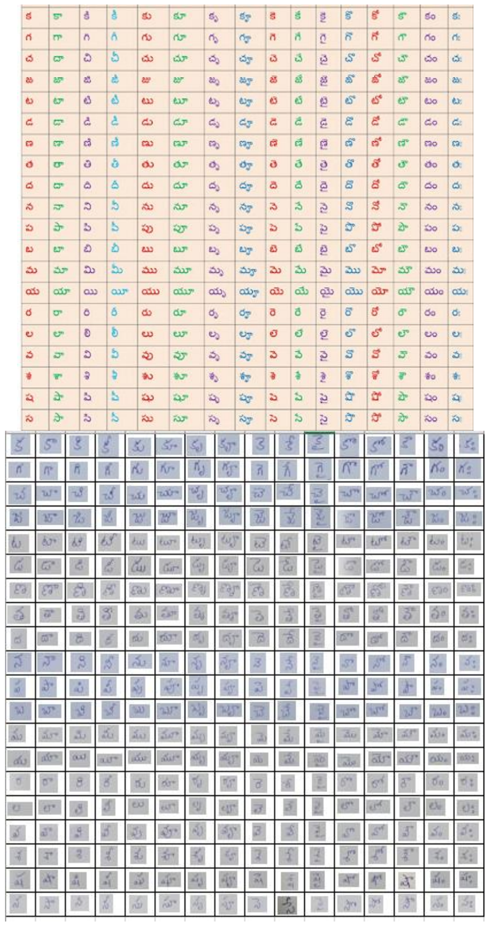


Fig. 3. Notations used in the Model (Both Online and Offline).

V. Requirements

The backend engine for the recommended model is Tensor Flow 2.2. CNN was implemented using the built-in dataset. These tests were run on a 64-bit operating system with an Intel i7-4770 CPU running at 4.00GHz and 16GB of RAM.

A. Results

The suggested model is applied to the pre-existing data set in the system. This research aims to develop a system for recognizing handwritten characters in Telugu, which makes extensive use of classification. Table I displays the model's layer-by-layer attributes. The proposed network will handle a total of 1,986,640 parameters.

Metrics including recall, precision, and accuracy [16] are used to assess the proposed system.

- Positive patterns that can be correctly anticipated from the totality of other positive patterns are "predicted with precision".

$$\Pr ecision = \frac{true\ positives}{true\ positives + false\ positives}$$

- Recall computes the fraction of positive patterns that are appropriately classified.

$$\operatorname{Re} call = \frac{true\ positives}{true\ positives + false\, negatives}$$

- In order to measure the model's classification accuracy, the percentage of accurate predictions is calculated. The formula provided may be used to figure this out.

$$Accuracy = \frac{true\ positives + true\, negatives}{true\, examples}$$

Tables II to V illustrate the results of different optimizers such as Adam, SGD and RMSprop with all the performance metrics. When compared to Adam, and SGD the RMSprop optimizer generated superior results. When RMSprop is employed, the learning rate is enhanced since it takes huge horizontal steps and converges faster. The loss function quantifies how close the network's predicted output and given ground truth labels are through forward propagation. The confusion matrices of different optimizers are shown in Fig. 4 to 6.

TABLE I. Parameters of CNN Model

Layer	Output Shape	Parameters
conv2d_16 (Conv2D)	68, 68, 256	2560
activation_21 (Activation)	68, 68, 256	0
max_pooling2d_	34, 34, 256	0
conv2d_17 (Conv2D)	32, 32, 256	590080
activation_22 (Activation)	32, 32, 256	0
max_pooling2d_	16, 16, 256	0
conv2d_18 (Conv2D)	14, 14, 256	590080
activation_23(Activation)	14, 14, 256	0
max_pooling2d_	7, 7, 256	0
flatten_6 (Flatten)	12544	0
dense_11 (Dense)	64	802880
dropout_6 (Dropout)	64	0
dense_12 (Dense)	16	1040
activation_24(Activation)	16	0

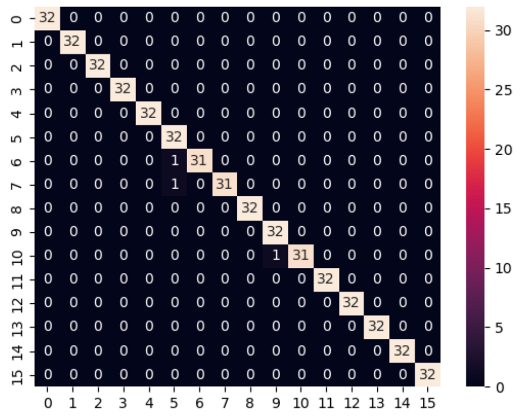


Fig. 4. Confusion Matrix with RMSprop.

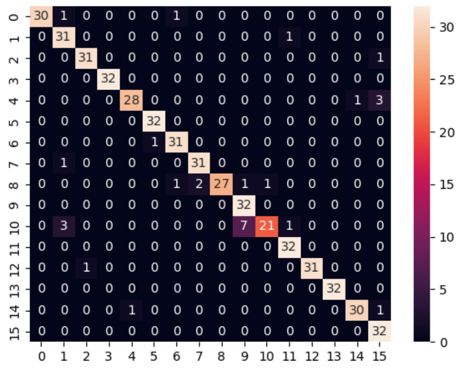


Fig. 5. Confusion Matrix with SDG.

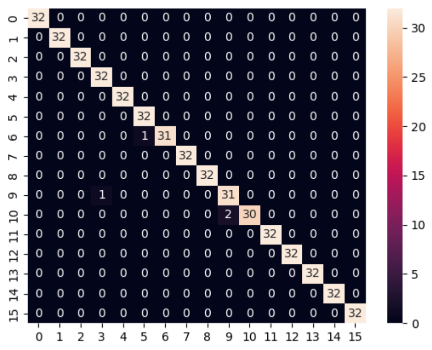


Fig. 6. Confusion Matrix with ADAM.

TABLE II. TRAINING AND TESTING ACCURACIES USING THE THREE OPTIMIZERS

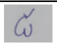
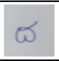
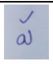
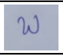
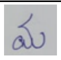
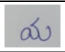
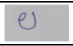
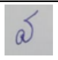
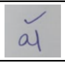
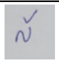
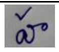
Guninth alu	**Training Accuracy**			**Testing Accuracy**		
	Adam Optimi zer	**RMSpr op Optimi zer**	**SGD Optimi zer**	**Adam Optimi zer**	**RMSpr op Optimi zer**	**SGD Optimi zer**
	94.5	96	93.8	84.63	89	83.21
	94.55	96.02	94.1	84.82	89.05	83.36
	94.6	96.1	94.21	84.9	90.02	83.39
	94.2	96.06	82.21	84.7	89.5	83.29
	92.6	94.29	81.64	83.5	88.4	82.06
	92.42	94.2	80.64	82.4	87.21	80.62
	95	96.5	93.25	85	91.2	83.12
	93.3	93.4	91.6	83.22	87.4	81.62
	95.2	97	94.32	85.3	90.2	83.46
	93.3	94.9	91.52	83.22	87.4	81.44
	95.2	96.2	93.89	85.2	89.64	83.62
	95.72	96.26	94.12	85.63	90.05	83.78
	94.41	95.62	92.98	84.32	88.64	82.36
	94.64	95.71	93.02	84.41	88.73	82.54
	94.2	94.9	92.91	84.21	88.32	82.25
	94.3	95.2	93.61	84.33	88.42	82.35
	95.62	96.22	94.01	85.62	89.21	83.14
	95.63	96.34	94.31	85.72	89.32	83.28
	95.68	96.41	94.65	85.91	89.64	83.36
	96.22	96.76	94.91	86.45	89.84	84.31
	96.34	96.82	93.8	86.54	90.01	83.21

TABLE III. PRECISION USING THE THREE OPTIMIZERS

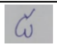
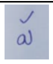
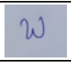
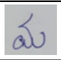
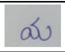
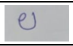
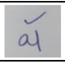
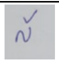
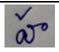
Guninthalu	**Precision**		
	Adam Optimizer	**RMSprop Optimizer**	**SGD Optimizer**
	0.835	0.895	0.821
	0.861	0.902	0.846
	0.87	0.912	0.851
	0.84	0.906	0.83
	0.828	0.892	0.819
	0.824	0.872	0.808
	0.841	0.91	0.824
	0.83	0.875	0.811
	0.862	0.882	0.842
	8.83	0.875	0.85
	0.842	0.912	0.83
	0.876	0.923	0.852
	0.824	0.884	0.806
	0.856	0.891	0.841
	0.826	0.872	0.802
	0.831	0.884	0.811
	0.846	0.916	0.831
	0.859	0.922	0.842
	0.869	0.934	0.851
	0.876	0.954	0.859
	0.885	0.961	0.821

TABLE IV. RECALL USING THE THREE OPTIMIZERS

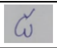
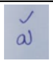
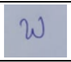
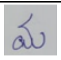
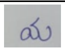
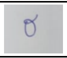
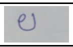
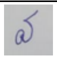
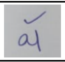
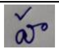
Guninthalu	**Recall**		
	Adam Optimizer	**RMSprop Optimizer**	**SGD Optimizer**
	0.812	0.89	0.795
	0.836	0.893	0.821
	0.842	0.897	0.837
	0.821	0.894	0.811
	0.819	0.881	0.809
	0.804	0.864	0.782
	0.826	0.916	0.812
	0.806	0.864	0.783
	0.829	0.871	0.816
	0.806	0.864	0.772
	0.824	0.912	0.804
	0.841	0.925	0.815
	0.808	0.884	0.794
	0.824	0.89	0.813
	0.824	0.884	0.818
	0.831	0.889	0.824
	0.836	0.91	0.827
	0.846	0.921	0.831
	0.854	0.933	0.842
	0.862	0.941	0.851
	0.868	0.948	0.795

TABLE V. LOSS USING THE THREE OPTIMIZERS

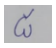
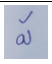
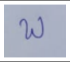
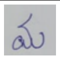
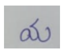
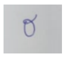
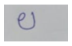
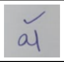
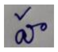
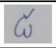
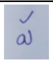
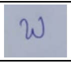
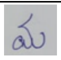
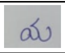
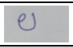
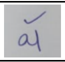
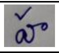
Guninthalu	**Loss**		
	Adam Optimizer	**RMSprop Optimizer**	**SGD Optimizer**
	0.63	0.39	0.68
	0.68	0.41	0.71
	0.71	0.5	0.74
	0.64	0.44	0.67
	0.6	0.4	0.62
	0.54	38	0.56
	0.7	0.47	0.72
	0.59	0.39	0.61
	0.68	0.46	0.71
	0.59	0.39	0.62
	0.65	0.41	0.68
	0.71	0.45	0.74
	0.59	0.37	0.61
	0.61	0.42	0.64
	0.58	0.36	0.6
	0.6	0.38	0.62
	0.65	0.41	0.67
	0.71	0.42	0.74
	0.75	0.44	0.77
	0.78	0.47	0.81
	0.79	0.51	0.68

TABLE VI. TOTAL PRECISION, RECALL, ACCURACY AND F1 SCORE WITH ADAM, RMSPROP AND SDG OPTIMIZERS

Optimizer	**Total Precision**	**Total Recall**	**Accuracy**	**F1 Score**
ADAM	0.9924	0.9921	99.21	0.9923
RMSprop	0.9944	0.9941	99.41	0.9942
SGD	0.948	0.943	94.33	0.945

TABLE VII. DETAILS OF VARIOUS GUNINTHALU OF PROPOSED DATABASE

Hand Written Characters	**Train-Test Split**	**No of Train Samples**	**No of Test Samples**
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104

	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104
	80-20	220416	55104

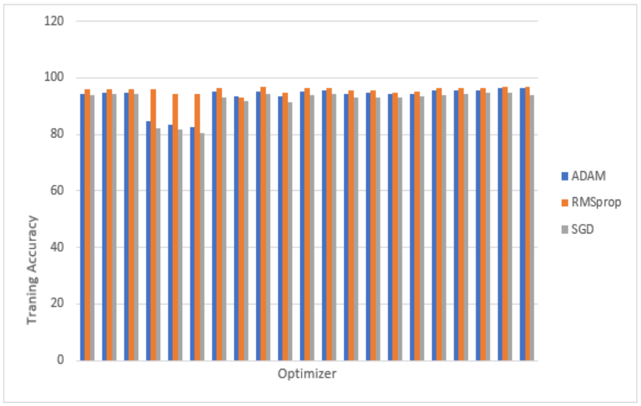


Fig. 7. Training Accuracy with ADAM, RMSprop & SGD Optimizer on 21 Guninthalu.

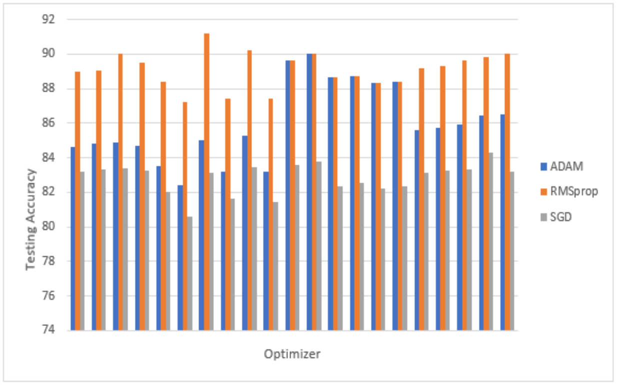


Fig. 8. Testing Accuracy with ADAM, RMSprop & SGD Optimizer on 21 Guninthalu.

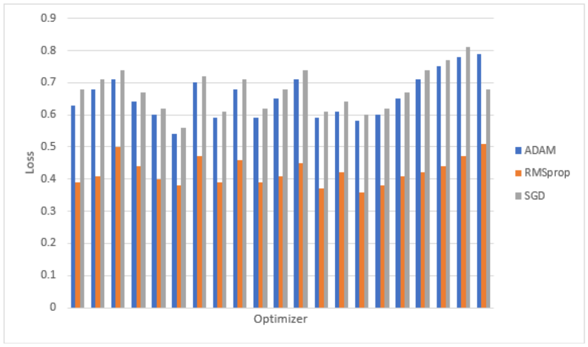


Fig. 9. Loss with ADAM, RMSprop & SGD Optimizer on 21 Guninthalu.

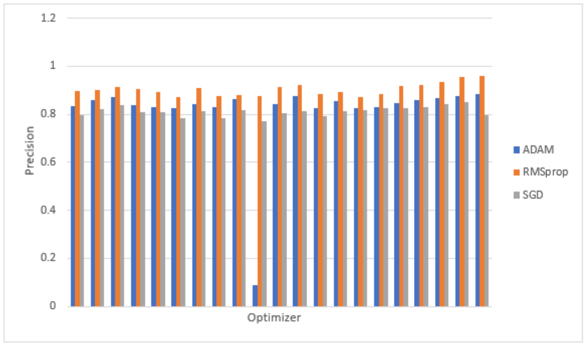


Fig. 10. Precision with ADAM, RMSprop & SGD Optimizer on 21 Guninthalu.

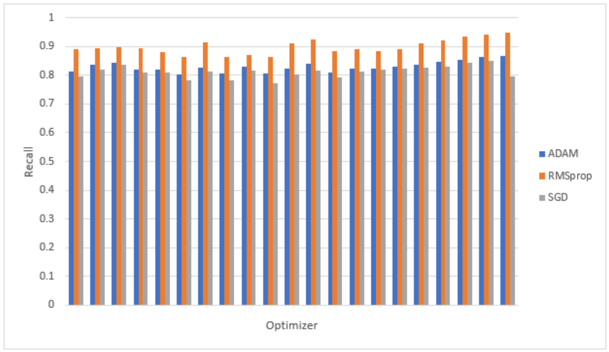


Fig. 11. Recall with ADAM, RMSprop & SGD Optimizer on 21 Guninthalu.

The training, testing accuracy, loss, precision, and recall with Adam optimizer, RMSprop optimizer, and SDG optimizer are shown in Fig. 7 to 11. The results show that the RMSprop optimizer has a higher testing accuracy than the ADAM and SGD optimizers. The mean values acquired with all the guninthalu are used to determine the overall training, testing accuracy, precision, and recall. Tables III to VI shows that when compared to ADAM and SGD optimizers, the RMSprop optimizer performs well on all guninthalu.

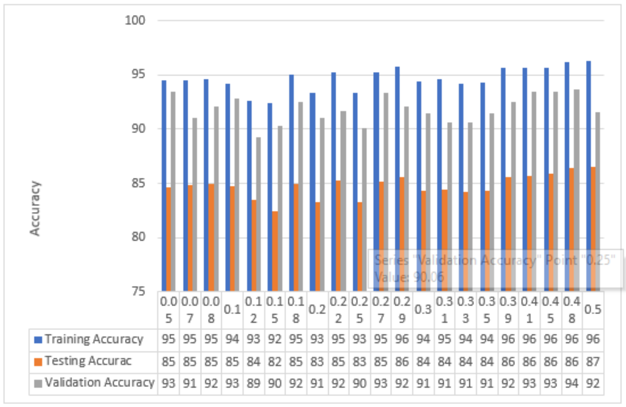
There are 820 different handwritten characters for each gunintham used to train the model. Each of the 21 guninthalu contains 16 characters, for a total of 21X16X820 = 275520 handwritten characters, which are used to train the model as shown in Table VII. After training, the model is checked using a 20% data mean and 55104 characters. Accuracy, Precision, Loss, and Recall are used to evaluate the model after it has been evaluated on 55104 data points.

B. Augmentation

Data augmentation is a method of artificially creating fresh training data from existing data. This is performed by transforming examples from the training data into fresh and unique training examples using domain-specific techniques.

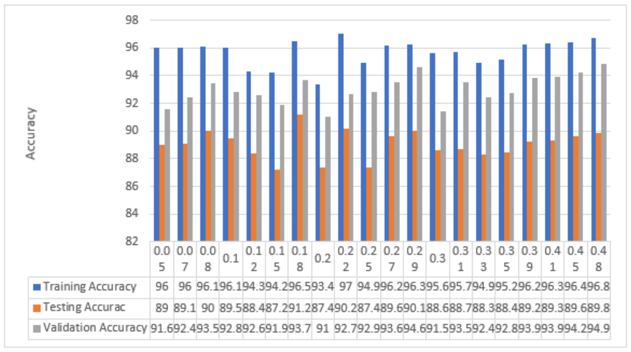
C. Dropout

Dropout is a technique for preventing overfitting in a model. Dropout operates by setting the outgoing edges of hidden units at random. As the augmentation factor is increased, the validation accuracy improves. The test accuracy increased to 90.12 with ADAM optimizer, 94.26 with RMSprop, and 86.72 with SGD optimizer after data augmentation as shown in Fig. 12 to 14.



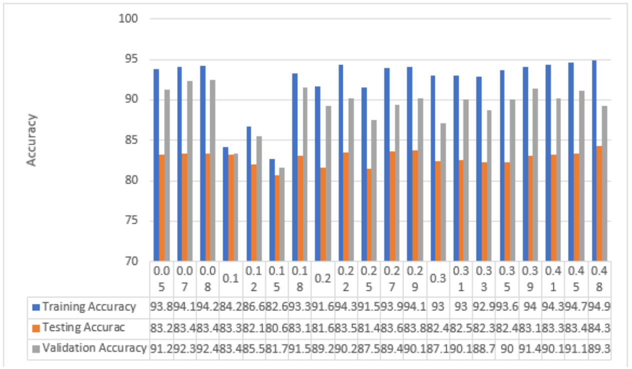
	0.05	0.07	0.08	0.1	0.12	0.15	0.18	0.2	0.22	0.25	0.27	0.29	0.3	0.31	0.33	0.35	0.39	0.41	0.45	0.48	0.5
■ Training Accuracy	95	95	95	94	93	92	95	93	95	93	95	96	94	95	94	94	96	96	96	96	96
■ Testing Accurac	85	85	85	85	84	82	85	83	85	83	85	86	84	84	84	84	86	86	86	86	87
■ Validation Accuracy	93	91	92	93	89	90	92	91	92	90	93	92	91	91	91	91	92	93	93	94	92

Fig. 12. Dropout in Layers with ADAM Optimizer.



	0.05	0.07	0.08	0.1	0.12	0.15	0.18	0.2	0.22	0.25	0.27	0.29	0.3	0.31	0.33	0.35	0.39	0.41	0.45	0.48
■ Training Accuracy	96	96	96.1	96.1	94.3	94.2	96.5	93.4	97	94.9	96.2	96.3	95.6	95.7	94.9	95.2	96.2	96.3	96.4	96.8
■ Testing Accurac	89	89.1	90	89.5	88.4	87.2	91.2	87.4	90.2	87.4	89.6	90.1	88.6	88.7	88.3	88.4	89.2	89.3	89.6	89.8
■ Validation Accuracy	91.6	92.4	93.5	92.8	92.6	91.9	93.7	91	92.7	92.9	93.6	94.6	91.5	93.5	92.4	92.8	93.9	93.9	94.2	94.9

Fig. 13. Dropout in Layers with RMSprop Optimizer.



	0.05	0.07	0.08	0.1	0.12	0.15	0.18	0.2	0.22	0.25	0.27	0.29	0.3	0.31	0.33	0.35	0.39	0.41	0.45	0.48
■ Training Accuracy	93.8	94.1	94.2	84.2	86.6	82.6	93.3	91.6	94.3	91.5	93.9	94.1	93	93	92.9	93.6	94	94.3	94.7	94.9
■ Testing Accurac	83.2	83.4	83.4	83.3	82.1	80.6	83.1	81.6	83.5	81.4	83.6	83.8	82.4	82.5	82.3	82.4	83.1	83.3	83.4	84.3
■ Validation Accuracy	91.2	92.3	92.4	83.4	85.5	81.7	91.5	89.2	90.2	87.5	89.4	90.1	87.1	90.1	88.7	90	91.4	90.1	91.1	89.3

Fig. 14. Dropout in Layers with SGD Optimizer.

VI. Conclusion

With CNN, our goal is to improve the quality of handwritten Telugu gunintham identification. The contributions of Adam, RMSprop, and SGD were critical in improving the model's accuracy and performance. The dropout and activation functions ReLu and Sigmoid are used to prevent overfitting. In CNN models, RMSprop optimizer outperformed Adam and SGD in terms of accuracy. This model might be adjusted to enhance the identification of handwritten Telugu characters. In the future, we want to continue experimenting with new approaches to improve the identification of Telugu handwritten guninthalu.

References

[1] Kou, F., Du, J., He, Y., & Ye, L. "Social Network Search Based on Semantic Analysis and Learning." CAAI Transactions on Intelligence Technology. 1 (2016) 293-302.

[2] Srinivas, S., Sarvadevabhatla, R. K., Mopuri, K. R., Prabhu, N., Kruthiventi, S. S., &Babu, R. V. (2016) "A taxonomy of deep convolutional neural nets for computer vision." https://doi.org/10.3389/frobt.2015.00036.

[3] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., &Torralba, A. (2014) "Object detectors emerge in deep scene cnns." https://arxiv.org/abs/1412.6856.

[4] Wang, Y., & Wu, Y. "Scene Classification with Deep Convolutional Neural Networks." (2014).

[5] Artha Andriyanto, Antoni Wibowo NS Norhaslinda Zainal Abidin "Sectoral Stock Prediction Using Convolutional Neural Networks with Candlestick Patterns as input Images" International Journal of Emerging Trends in Engineering Research 8(2020) 2249-2252.

[6] P.V. Ramana Murthy, Ch. G.V.N. Prasad "Recognition of Online Handwritten Telugu Letters For Different Domains And Organizations" Journal of Critical Reviews 6 (2019.) 33-40.

[7] P. Sujatha, D. Lalitha Bhaskari "Telugu and Hindi Script Recognition using Deep learning Techniques" International Journal of Innovative Technology and Exploring Engineering (IJITEE) 8 (2019) 1758-1764.

[8] Buddaraju Revathi, G.Naveen Kishore, V Dheera "A Survey On OCR For Telugu Language" International Journal Of Scientific & Technology Research 8 (2019) 559-562.

[9] B. Hari Kumar, P. Chitra "Survey Paper Of Script Identification Of Telugu Language Using OCR" International Journal of Electronics and Communication Engineering (IJECE) 8 (2019) 15-20.

[10] Chandra Prakash Konkimalla et al., Optical Character Recognition (OCR) for Telugu: Database, Algorithm and Application arXiv:1711.07245 (2018).

[11] Chirag I Patel, Ripal Patel, Palak Patel "Handwritten Character Recognition using Neural Network" International Journal of Scientific & Engineering Research 2 (2011) 1-6.

[12] A Ram Bharadwaj, A. Venugopal, Ch. Surya Kiran, M. V. Nageswara Rao "Telugu text extraction and recognition using convolutional and recurrent neural networks" International Journal of Engineering and Advanced Technology (IJEAT) 8 (2019) 1449-1451.

[13] N. Shaffi and F. Hajamohideen, "uTHCD: A New Benchmarking for Tamil Handwritten OCR," in IEEE Access, vol. 9, pp. 101469-101493, 2021, doi: 10.1109/ACCESS.2021.3096823.

[14] Richard HR Hahnloser, Rahul Sarpeshkar, Misha A Mahowald, Rodney J Douglas,and H Sebastian Seung "Digital selection and analogue amplification coexistin a cortex-inspired silicon circuit" Nature 405 (2000), 947.

[15] Diederik P. Kingma, Jimmy Ba, "Adam: A Method for Stochastic Optimization", arXiv:1412.6980.

[16] Praveen Kumar Kollu, R. Satya Prasad "Intrusion Detection System Using Recurrent Neural Networks and Attention Mechanism", International Journal of Emerging Trends in Engineering Research 7(2019) 178-182.